

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

Java™ Web Services
by David A. Chappell and Tyler Jewell

Copyright © 2002 O'Reilly & Associates, Inc. All rights reserved.
Printed in the United States of America.

Published by O'Reilly & Associates, Inc., 1005 Gravenstein Highway North, Sebastopol, CA
95472.

O'Reilly & Associates books may be purchased for educational, business, or sales promotional
use. Online editions are also available for most titles (my.safaripublishing.com). For more information
contact our corporate/institutional sales department: (800) 998-9938 or corporate@oreilly.com.

Editor: Mike Loukides
Production Editor: Ann Schirmer
Cover Designer: Emma Colby
Interior Designer: Melanie Wang
Printing History:
March 2002: First Edition.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered
trademarks of O'Reilly & Associates, Inc. Java™ and all Java-based trademarks and logos are
trademarks or registered trademarks of Sun Microsystems, Inc., in the United States and other
countries. O'Reilly & Associates, Inc. is independent of Sun Microsystems. Many of the
designations used by manufacturers and sellers to distinguish their products are claimed as
trademarks. Where those designations appear in this book, and O'Reilly & Associates, Inc. was
aware of a trademark claim, the designations have been printed in caps or initial caps. The
association between the image of a European ibex and Java web services is a trademark of O'Reilly
& Associates, Inc.

While every precaution has been taken in the preparation of this book, the publisher and author
assume no responsibility for errors or omissions, or for damages resulting from the use of the
information contained herein.

ISBN: 0-596-00269-6
[M]

Table of Contents

Preface	vii
1. Welcome to Web Services	1
What Are Web Services?	1
Web Services Adoption Factors	7
Web Services in a J2EE Environment	10
What This Book Discusses	11
2. Inside the Composite Computing Model	13
Service-Oriented Architecture	14
The P2P Model	23
3. SOAP: The Cornerstone of Interoperability	25
Simple	25
Object	26
Access	27
Protocol	27
Anatomy of a SOAP Message	28
Sending and Receiving SOAP Messages	32
The Apache SOAP Routing Service	45
SOAP with Attachments	49
4. SOAP-RPC, SOAP-Faults, and Misunderstandings	54
SOAP-RPC	54
Error Handling with SOAP Faults	63
SOAP Intermediaries and Actors	69

A web service has special behavioral characteristics:

XML-based

By using XML as the data representation layer for all web services protocols and technologies that are created, these technologies can be interoperable at their core level. As a data transport, XML eliminates any networking, operating system, or platform binding that a protocol has.

Loosely coupled

A consumer of a web service is not tied to that web service directly; the web service interface can change over time without compromising the client's ability to interact with the service. A tightly coupled system implies that the client and server logic are closely tied to one another, implying that if one interface changes, the other must also be updated. Adopting a loosely coupled architecture tends to make software systems more manageable and allows simpler integration between different systems.

Coarse-grained

Object-oriented technologies such as Java expose their services through individual methods. An individual method is too fine an operation to provide any useful capability at a corporate level. Building a Java program from scratch requires the creation of several fine-grained methods that are then composed into a coarse-grained service that is consumed by either a client or another service. Businesses and the interfaces that they expose should be coarse-grained. Web services technology provides a natural way of defining coarse-grained services that access the right amount of business logic.

Ability to be synchronous or asynchronous

Synchronicity refers to the binding of the client to the execution of the service. In synchronous invocations, the client blocks and waits for the service to complete its operation before continuing. Asynchronous operations allow a client to invoke a service and then execute other functions. Asynchronous clients retrieve their result at a later point in time, while synchronous clients receive their result when the service has completed. Asynchronous capability is a key factor in enabling loosely coupled systems.

Supports Remote Procedure Calls (RPCs)

Web services allow clients to invoke procedures, functions, and methods on remote objects using an XML-based protocol. Remote procedures expose input and output parameters that a web service must support. Component development through Enterprise JavaBeans (EJBs) and .NET Components has increasingly become a part of architectures and enterprise deployments over the past couple of years. Both technologies are distributed and accessible through a variety of RPC mechanisms. A web service supports RPC by providing services of its own, equivalent to those of a traditional component, or by translating incoming invocations into an invocation of an EJB or a .NET component.

Supports document exchange

One of the key advantages of XML is its generic way of representing not only data, but also complex documents. These documents can be simple, such as when representing a current address, or they can be complex, representing an entire book or RFQ. Web services support the transparent exchange of documents to facilitate business integration.

The Major Web Services Technologies

Several technologies have been introduced under the web service rubric and many more will be introduced in coming years. In fact, the web service paradigm has grown so quickly that several competing technologies are attempting to provide the same capability. However, the web service vision of seamless worldwide business integration is not feasible unless the core technologies are supported by every major software company in the world.

Over the past two years, three primary technologies have emerged as worldwide standards that make up the core of today's web services technology. These technologies are:

Simple Object Access Protocol (SOAP)

SOAP provides a standard packaging structure for transporting XML documents over a variety of standard Internet technologies, including SMTP, HTTP, and FTP. It also defines encoding and binding standards for encoding non-XML RPC invocations in XML for transport. SOAP provides a simple structure for doing RPC: document exchange. By having a standard transport mechanism, heterogeneous clients and servers can suddenly become interoperable. .NET clients can invoke EJBs exposed through SOAP, and Java clients can invoke .NET Components exposed through SOAP.

Web Service Description Language (WSDL)

WSDL is an XML technology that describes the interface of a web service in a standardized way. WSDL standardizes how a web service represents the input and output parameters of an invocation externally, the function's structure, the nature of the invocation (in only, in/out, etc.), and the service's protocol binding. WSDL allows disparate clients to automatically understand how to interact with a web service.

Universal Description, Discovery, and Integration (UDDI)

UDDI provides a worldwide registry of web services for advertisement, discovery, and integration purposes. Business analysts and technologists use UDDI to discover available web services by searching for names, identifiers, categories, or the specifications implemented by the web service. UDDI provides a structure for representing businesses, business relationships, web services, specification metadata, and web service access points.

Individually, any one of these technologies is only evolutionary. Each provides a standard for the next step in the advancement of web services, their description, or their discovery. However, one of the big promises of web services is seamless, automatic business integration: a piece of software will discover, access, integrate, and invoke new services from unknown companies dynamically without the need for human intervention. Dynamic integration of this nature requires the combined involvement of SOAP, WSDL, and UDDI to provide a dynamic, standard infrastructure for enabling the dynamic business of tomorrow. Combined, these technologies are revolutionary because they are the first standard technologies to offer the promise of a dynamic business. In the past, technologies provided features equivalent to SOAP, WSDL, and UDDI in other languages, but they weren't supported by every major corporation and did not have a core language as flexible as XML.

Figure 1-1 provides a diagram that demonstrates the relationship between these three technologies.

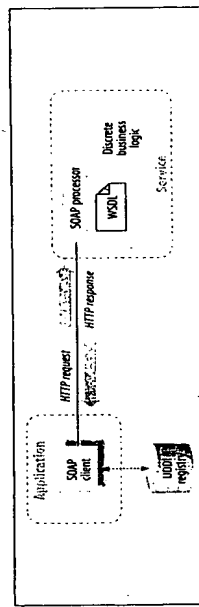


Figure 1-1. Simple web service interaction

The relationship between these pieces (SOAP, WSDL, and UDDI) can be described as follows: an application acting in the role of a web services client needs to locate another application or a piece of business logic located somewhere on the network. The client queries a UDDI registry for the service either by name, category, identifier, or specification supported. Once located, the client obtains information about the location of a WSDL document from the UDDI registry. The WSDL document contains information about how to contact the web service and the format of request messages in XML schema. The client creates a SOAP message in accordance with the XML schema found in the WSDL and sends a request to the host (where the service is).

Service-Oriented Architecture in a Web Services Ecosystem

The web services model lends itself well to a highly distributed, service-oriented architecture (SOA). A web service may communicate with a handful of standalone processes and functions or participate in a complicated, orchestrated business process. A web service can be published, located, and invoked within the enterprise, or anywhere on the Web.

As illustrated in Figure 1-2, a service might be simple and discrete, such as an international currency conversion service. It may also be a whole suite of applications representing an entire business function, such as an auto insurance claims processor. At the mass-consumer market, web services may provide something like a restaurant finder application for a handheld device that knows who and where you are. It could also take the form of an application that participates in an exchange between a business entity and its suppliers.

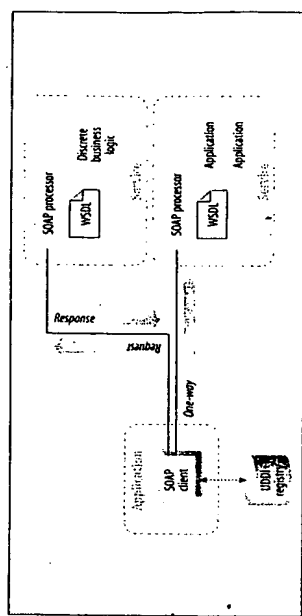


Figure 1-2. Discrete components in a web services architecture

Whether a service is implemented as a fine-grained component performing a discrete operation or as an application suite exposing an entire business function, each can be considered a self-contained, self-describing, modular unit that participates in a larger ecosystem. As illustrated in Figure 1-3, a web service can access and encapsulate other web services to perform its function. For example, a portal such as *www.boston.com* may have a restaurant finder application that is exposed as a web service. The restaurant finder service may in turn access Mapquest as a web service in order to get directions.

Eventually, these small ecosystems can all be combined into a larger, more complicated, orchestrated business macrocosm.

A service-oriented architecture may be intended for use across the public Internet, or built strictly for private use within a single business or among a finite set of established business partners.

Practical Applications for Web Services

Because of the cross-platform interoperability promised by SOAP and web services, we can provide practical business solutions to problems that, until now, have only been a dream of distributed-computing proponents.

via another web service. If all pharmacies in the world standardized a communication protocol for accepting prescriptions, the doctor could write you a subscription for any pharmacy that you selected. The pharmacy would be able to fulfill the prescription immediately and have it prepared for you when you arrive or couriered to your residence.

This model can be extended further. If the interfaces used between doctors and pharmacies are standardized using web services, a portal broker could act as an intermediary between doctors and pharmacies providing routing information for requests and better meet the needs of individual consumers. For example, a patient may register with an intermediary and specify that he wants to use generic drugs instead of expensive brand names. An intermediary can intercept the pharmaceutical web service request and transform the request into a similar one for the generic drug equivalent. The intermediary exposes web services to doctors and pharmacies (in both directions) and can handle issues such as security, privacy, and nonrepudiation.

Web Services Adoption Factors

Web services are new technologies and require a paradigm shift. The adoption of web services is directly impacted by the adoption of the paradigm of web services development.

A paradigm shift can happen quickly in a large wave, when suddenly the whole world is doing something differently, and no one notices how and when it happened until after the fact. An example of such a shift is the World Wide Web phenomenon that began around 1995. The combination of HTML, HTTP, and the CGI programming model is not the most efficient way to accomplish the services offered by these technologies, yet the CGI model gained widespread grassroots acceptance because it was simple and easy to adopt.

The acceptance of CGI started the wave. To become a lasting paradigm shift, the model of web-based business needed broader acceptance among corporate IT and industry leaders. This acceptance was encouraged by continuing standards development within W3C and IETF and through continuing technology innovations such as NSAPI, NSAPI, Java Servlets, and application servers. Eventually, high-level architectures and infrastructures such as .NET and J2EE were created to hold everything together.

Unlike the initial adoption of the Web, which was driven by grass-roots demand, the adoption of web services will be driven downward by corporations. It's still a paradigm shift, but it's likely to move more slowly. The adoption of the fax machine provides a good analogy. Because fax machines were initially large expensive devices, they were adopted first by large businesses as a way to communicate between their offices. As more companies bought fax machines, they became important for business-to-business communications. Today, fax machines are nearly ubiquitous—

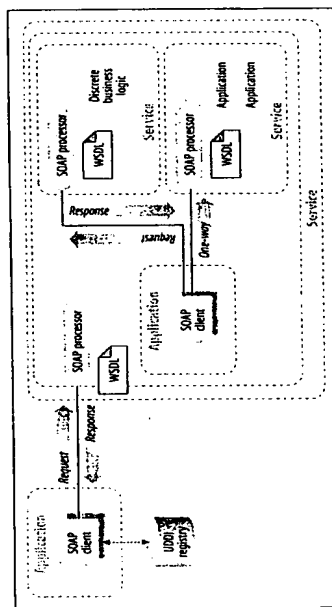


Figure 1-3. Web services within a larger ecosystem

It's easy to see the use for simple, discrete web services such as a currency conversion service that converts dollars to Euros or a natural language translation service that converts English to French. Today, web sites such as www.xmethods.com are dedicated to hosting simple web services.

This scenario becomes more exciting when we see real companies using web services to automate and streamline their business processes. Let's use the concept of a Business-to-Consumer (B2C) portal. Web-based portals, such as those used by the travel industry, often combine the offerings of multiple companies' products and services and present them with a unified look and feel to the consumer accessing the portal. It's difficult to integrate the backend systems of each business to provide the advertised portal services reliably and quickly.

Web services technology is already being used in the integration between Dollar Rent A Car Systems, Inc. and Southwest Airlines Co. Dollar uses the Microsoft SOAP Toolkit to integrate its online booking system with Southwest Airlines Co.'s site. Dollar's booking system runs on a Sun Solaris server, and Southwest's site runs on a Compaq OpenVMS server. The net result (no pun intended) is that a person booking a flight on Southwest Airline's web site can reserve a car from Dollar without leaving the airline's site. The resulting savings for Dollar are a lower cost per transaction. If the booking is done online through Southwest and other airline sites, the cost per transaction is about \$1.00. When booking through traditional travel agent networks this cost can be up to \$5.00 per transaction.

The healthcare industry provides many more scenarios in which web services can be put to use effectively. A doctor carrying a handheld device can access your records, health history, and your preferred pharmacy using a web service. The doctor can also write you an electronic prescription and send it directly to your preferred pharmacy using an electronic prescription service.

you can fax in your pizza order. We expect to see the same trend in web services. They will be used first for internal business communications before they become part of everyday life. In all cases, though—the rapid adoption of the Web, the slower adoption of the fax machine, and the current adoption of web services—the same factor has enabled the paradigm shift. That factor is a standards communications mechanism. Whether the standard be the phone line and modem protocols, the TCP/IP stack and HTTP (together with the phone line and modem protocols), or the web service protocols, standards have been, and continue to be, the key factor in enabling the acceptance of new technologies.

Industry Drivers

Many tangible drivers make web services technology attractive, both from a business and a technical perspective. Classic Enterprise Application Integration (EAI) problems require applications to integrate and interoperate. Even within a particular business unit, there exist islands of IT infrastructure. For example, a Customer Relationship Management (CRM) system may have no knowledge of how to communicate with anything outside of its own application suite. It may need to communicate with a third-party Sales Order system so it can know about new customers as soon as they place their first order.

Corporate acquisitions and mergers are also an issue. Entire parallel business application infrastructures have to be synchronized or merged. Business partners such as suppliers and buyers need to collaborate across corporate boundaries.

These EAI and B2B problems exist in abundance and are increasing exponentially. Every new deployed system becomes a legacy system, and any future integration with that system is an EAI or B2B problem. As the growth of integration problems and projects accelerates over the next couple of years, the standards-based approach that web services offer makes adopting web services technology an attractive option for companies that need to cost-effectively accomplish seamless system integration.

Lessons Learned from Recent History

Some industry analysts claim that the web service model is causing a paradigm shift that will change the way distributed computing is done forever. Others say that this model is just a fad that will go away soon. Currently, web services is still very much in the hype phase. Drawing parallels to other new technologies can teach us important lessons.

Other distributed-computing models have had an opportunity to garner universal acceptance and adoption, yet they have not. While these models offer great technical advantages for solving real problems, none have achieved the massive widespread adoption that their proponents had hoped for. This is largely due to their proprietary nature and the inevitable vendor lock-in. Though COM/DCOM had a widespread following, it could not permeate an enterprise because it was limited to

Microsoft platforms. CORBA was controlled by the OMG, a neutral standards body. However, software availability was a problem. There were really only two robust vendor implementations: Iona and Visigenic.

Forcing middleware infrastructure down the throats of other departments and business partners is not easy. Both CORBA and DCOM required that a piece of the vendor-supplied middleware be installed at every node of the system. You can't always force a business partner to install a piece of your software at their site for them to be able to participate in business transactions with your systems. Even within the four walls of an organization, agreeing upon and rolling out an enterprise-wide middleware solution is a huge, concerted effort. CORBA implementations eventually achieved cross-vendor interoperability, but by then it was too late; the wave had already passed.

Crossing corporate boundaries in a secure, reliable fashion is key. If you go back only as far as 1996 to 1997, you would have seen every trade magazine talking about a world of distributed CORBA objects happily floating around on the Internet, discovering one another dynamically and communicating through firewalls. Standards were proposed for firewall communications, and IIOP was going to be adopted by all major firewall vendors as a recognizable protocol. It just never happened—partly due to the aforementioned adoption problems and partly due to widespread adoption and general acceptance of HTTP as a firewall-friendly protocol.

Why Web Services, and Why Now?

What is so different about web services, and why are they poised for success, whereas other preceding technologies have failed to achieve widespread adoption? The answer lies in the challenge that every organization faces today: to create a homogeneous environment while still leveraging its core abilities and existing applications. IT needs a simple, platform-neutral way of communicating between applications.

For starters, XML is ideal for representing data. IT developers have had exposure to XML for a few years and they understand what it's good for. Even though the average IT developer hasn't yet become a walking XML parser, by now most developers understand the concepts behind XML and how it can be used.

Also, the base technologies of SOAP, WSDL, and UDDI are not themselves very exciting; they are just new dressings for the same old distributed-computing model. What draws people to them is the promise of what they enable. Finally, we have a platform-neutral communication protocol that provides interoperability and platform independence. A bidirectional conversation may occur between a Biztalk server and a set of hand-rolled Perl scripts. The Perl scripts may be simultaneously involved in a conversation with a set of applications held together by a J2EE-based application server or a message-oriented middleware (MOM) infrastructure. The minimum requirement is that each participant in the multiparty collaboration knows how to construct and deconstruct SOAP messages and how to send and receive HTTP transmissions.